# Notes on Project 3, Spring 2017:

## Tasks:

### Admin:

Github setup – Done

Presentation slides

### documentation

Specification of the system

## System Specification

### General Approach

Dr. Fofanov suggested that he would try a seed based approach if he were solving the problem, unlike our idea of storing all genomic sequences for a given TaxID into a single prefix tree. He seemed genuinely intrigued about the approach of using a prefix tree at this time. He tried it 15 years ago and it did not turn out well, but with advances in compute technology, it might be worth another try. But one suggestion he did have was to put as many sequences as possible into the tree, where the terminator string for a sequence is it’s TaxID. This would have the advantage of allowing us to simply traverse to the end of the tree to find all TaxID’s that match a given query (if it hits), and also store as many possible subjects into the data structure at a time so that the time spend loading/unloading data from memory would be minimize.

As such, the current general approach would be to store all of the ***subject*** genomic sequences into a prefix tree, and take the queries directly from the fasta file and search them 1-by-1, recording results. It may also be possible (and wise) to add parallelism such that multiple queries could be searched against the subjects in the tree at the same time. Options include using a CPU with multi-threading, or a GPU with block launching.

### Details

## Code: (below are some ideas about how to potentially organize the code)

### Philosophy

1. Use Libraries where possible. For example, finding a Prefix Tree library that already exists and has full code functionality will save time, assuming it is implemented correctly for our needs.
2. Use camelCase naming convention for functions and “\_” naming convention for variables?
3. Regard the header files as an interface. This allows all code authors to know what to expect from functions that they are not implementing, but will be using.

### Organization of code

FASTA class:

1. file reader
   1. read subject sequences into the tree
      1. params: file name, pointer to tree
      2. param: number of sequences to read (or total bases, to limit tree size)
      3. return: sequence, GI??, TaxID
   2. read query fragments into a data structure (for parallelism), or just run the query searches line-by-line from the file directly?

Tree class:

1. Data Structure design and methods
2. traversal/search (needs to start from all nodes in the tree)
   1. I think we need a suffix tree here, so we don’t have to search from every node in the tree. Wait, do suffix tree have the same problem?
3. seed generation?? (if we choose something seed-based)
4. fuzzy alignment logic

Reporting class:

- nicely report the binned data

Unit testing:

- all (or important) methods of the Tree class

Parallelism

- we should probably support threads at a minimum

- GPU threading would be nice on the K80 cluster, the problem lends itself well to parallelism

## Open Questions or concerns

How to deal with the non-ACGT letters?

Should we make the number of tolerated mismatches proportional to the length of the sequence? Actually, no, we are trying fixed-length queries against variable length subjects

Meeting with Dr. F:

* Parallelism should happen at the read level (thread out 10k reads to each CPU, with one tree loaded into memory
* Precomputing to store the DB on disc for quick load later
* GPUs not good for this problem, memory too small on GPU
* GPUs could work on seed-based
  + Precompute the locations of the seeds
  + Seed based needs to have very fast proc for finding the clustering, the pile-ups

2 problems for seed based

* + How to cluster these things quickly and start with best one
  + How do you know if the best hit is worth expansion(could run simulation here to find the probability) – need to do this to find out how many clustered seeds we need to even bother
  + Other problem is stopping condition (NW) expensive
* 85 distinct seeds (with k=16 seeds, 100 mer)
* Up to 3 mismatches

Hash table – Jiaming

File reader (this weekend) – Mitchell

Table contents -

Seed generation – Mohammed

Function to quickly return GI for a given index in the combined sequence

Alignment (NW) -

Algorithm to analyze clustering -

Simulation/optimization to determine min score (lowest priority)

Harvard – CS50 Youtube (basics)